6. 自动化构建

Jenkins是一个自动化构建工具，它可以帮助我们摆脱繁琐的部署过程，我们只需要在一开始配置好构建策略，以后部署只需要一键完成。

6.1 创建Jenkins容器

Jenkins采用Java开发，也需要Java环境，但我们使用Docker后，一切都采用容器化部署，Jenkins也不例外。

拉取镜像

这里我们使用Jenkins官方提供的镜像，大家只需执行如下命令拉取即可：

docker pull docker.io/jenkins/jenkins

启动容器

由于Jenkins运行在Tomcat容器中，因此我们将容器的8080端口映射到宿主机的10080端口上：

docker run --name jenkins -p 10080:8080 docker.io/jenkins/jenkins

初始化Jenkins

然后你需要访问IP:10080，Jenkins会带着你进行一系列的初始化设置，你只要跟着它一步步走就行了，比较傻瓜式。

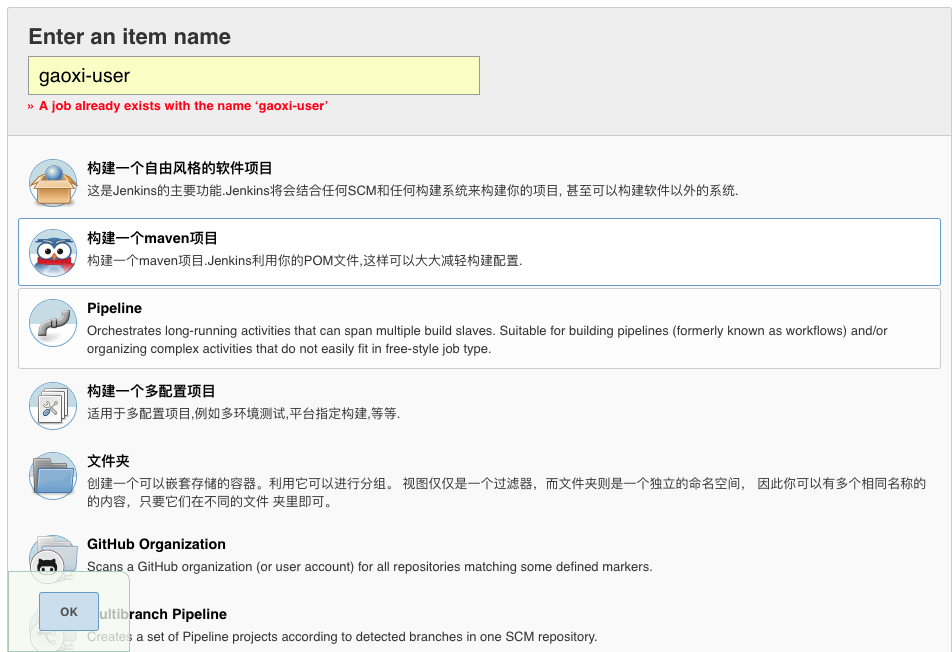
6.2 在Jenkins中创建项目

接下来我们要做的是，在Jenkins中为每一个服务创建一个项目，每个项目中定义了构建的具体流程。由于我们将整个项目分成了6个微服务，所以我们需要在Jenkins中分别为这6个服务创建项目。那句开始吧～

点击页面左侧的“新建”按钮：

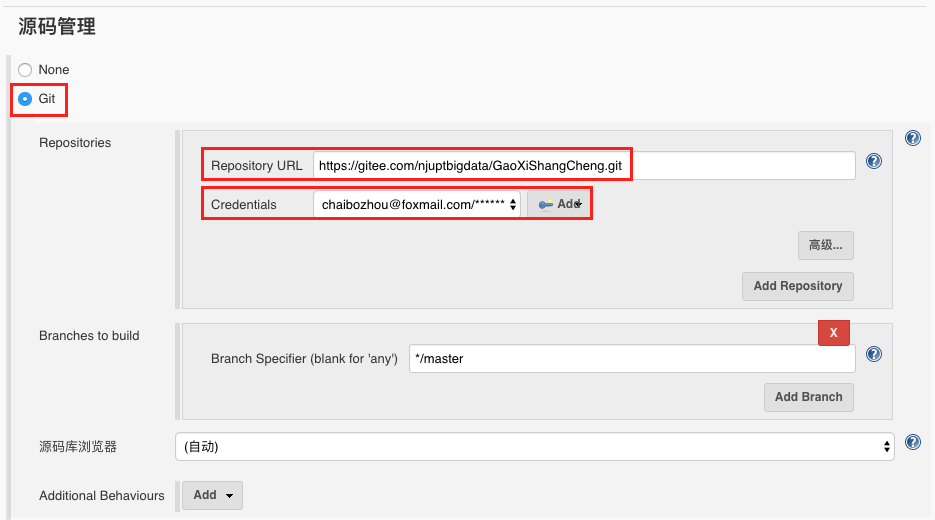


输入项目名称gaoxi-user，选择“构建一个Maven项目”，然后点击“OK”：



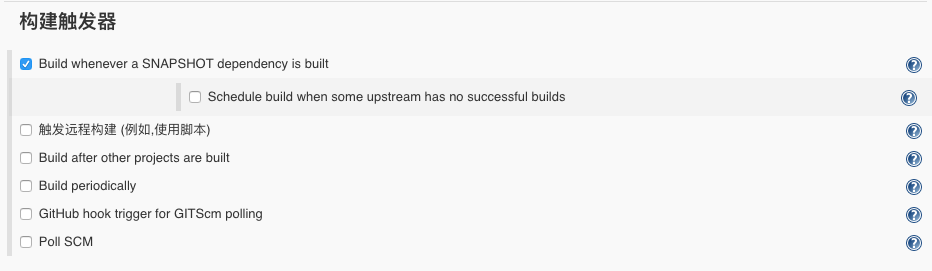
配置Git仓库

选择Git，然后输入本项目Git仓库的URL，并在Credentials中输入Git的用户名和密码，如下图所示：



构建触发器

选择第一项，如下图所示：



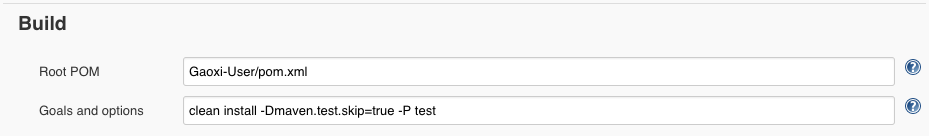
Pre Step

Pre Step会在正式构建前执行，由于所有项目都依赖于Gaoxi-Common-Service—Facade，因此在项目构建前，需要将它安装到本地仓库，然后才能被当前项目正确依赖。 因此，在Pre Step中填写如下信息：



Build

然后就是正式构建的过程，填写如下信息即可：



OK，Gaoxi-User的构建过程就配置完成了。当我们点击“立即构建”按钮时，Jenkins首先会从我们指定的Git仓库中拉取代码，然后执行Pre Step中的Maven命令，将Gaoxi-Common-Serivce-Facade打包安装到本地仓库。然后执行Build过程，将Gaoxi-User进行编译打包。 但此时Gaoxi-User仍然只是一个本地war包，并没有部署到Tomcat容器中，而我们采用了容器化部署后，Jenkins服务和Gaoxi-User服务并不在同一个Docker容器中，那么究竟该如何才能将Jenkins本地编译好的war包发送到Gaoxi-User容器中呢？这就需要使用Jenkins的一个插件——Deploy Plugin。

6.3 远程部署

下载插件

首先你需要下载Deploy Plugin，下载地址如下： <https://wiki.jenkins.io/display/JENKINS/Deploy+Plugin>

安装插件

在系统管理–>插件管理–>高级上传deploy.hpi进行安装。

在父项目的pom文件中增加远程部署插件：

为Tomcat设置用户名和密码

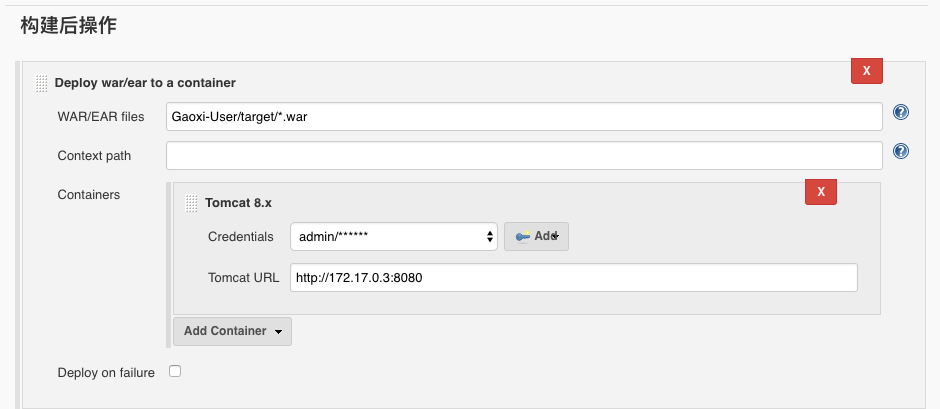
修改gaoxi-user容器中tomcat的tomcat-users.xml文件，增加tomcat的manager用户



注意：如果你使用了chaimm/tomcat镜像，那么其中Tomcat配置都已经完成，默认用户名：admin、默认密码：jishimen2019。强烈建议修改用户名和密码。

修改Jenkins中gaoxi-user的配置

在“构建后操作”中增加如下配置：



WAR/EAR files：表示你需要发布的war包

Containers：配置目标Tomcat的用户名和密码

7. Maven的profile功能

在实际开发中，我们的系统往往有多套环境构成，如：开发环境、测试环境、预发环境、生产环境。而不同环境的配置各不相同。如果我们只有一套配置，那么当系统从一个环境迁移到另一个环境的时候，就需要通过修改代码来更换配置，这样无疑增加了工作的复杂度，而且易于出错。但好在Maven提供了profile功能，能帮助我们解决这一个问题。

父项目的pom中添加profile元素

首先，我们需要在总pom的中添加多套环境的信息，如下所示：

父项目的pom中添加resource元素

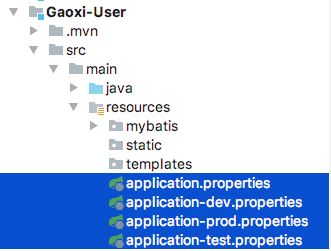
resource标识了不同环境下需要打包哪些配置文件。

父项目的pom中添加插件maven-resources-plugin

该插件用来在Maven构建时参数替换

在子项目中创建配置

分别为dev环境、test环境、prod环境创建三套配置，application.proerpties中存放公用的配置。



在application.properties中添加spring.profiles.active=@profileActive@

spring.profiles.active=@profileActive@

修改Jenkins的配置

在所有Jenkins中所有Maven命令的末尾添加 -P test，在打包的时候-P后面的参数将会作为@profileActive@的值传入系统中，从而根据该值打包相应的application-{profileActive}.properties文件。